DAHİLİ SINIFLAR (INNER CLASSES)

Öğr.Gör. Dr. Murat ASLANYÜREK

DAHİLİ SINIFLARA GİRİŞ

- Java dilinde diğer dillerdeki gibi çoklu kalıtım yoktur. Java çoklu kalıtımı arayüz(interface) ve dahili sınıflar(inner classes) ile sağlar. Dahili sınıfları sınıf içerisinde tanımlanmış sınıf olarak belirtebiliriz.
- Dahili sınıflar 3 ana grupta incelenir
  - 1.Dahili Üye Sınıflar(static üye sınıflar, static olmayan üye sınıflar)
  - 2.Yerel Sınıflar
  - 3.İsimsiz Sınıflar

1. DAHİLİ ÜYE SINIFLAR

- Dahili üye sınıflar sınıf içerisinde tanımlanmış sınıflardır. Dahili üye sınıflar sayesinde parçalar bir araya gelerek bütünü oluşturabilirler.

```
1  package Dahili_Siniflar;
2
3  public class bilgisayar {
4      public class anakart{
5          //1. dahili üye sınıf
6      }
7      public class islemci{
8          //2. dahili üye sınıf
9      }
10     public class RAM_Bellek{
11         //3. dahili üye sınıf
12     }
13 public static void main(String[] args){
14     bilgisayar.RAM_Bellek RAM = new bilgisayar().new RAM_Bellek();
15
16   }
17 }
```

Not: Dahili sınıf nesnesi Oluşturabilmek için üst sınıf kullanılır.

Örnek(*islemYap.java*)

```java
package Dahili_Siniflar;

public class islemYap {
    public class DahiliUyeSinif {
        private void metot1() {
            System.out.println("Dahili Üye Sınıfın Metotu");
        }
        public void metot2(int a, int b) {
            System.out.println(a*b);
        }
    }
    public static void main(String[] args){
        islemYap.DahiliUyeSinif n1= new islemYap().new DahiliUyeSinif();
        n1.metot1();
        n1.metot2(4,8);
    }

}
```

Console ⊠ Problems Debug Shell

<terminated> islemYap [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (28 Mar 2020 22:11:

```
Dahili Üye Sınıfın Metotu
32
```

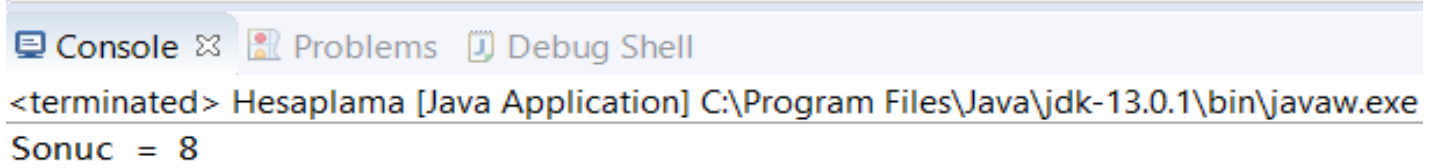
Örnek(*Hesaplama.java*)

```java
package Dahili_Siniflar;

public class Hesaplama {

    public class Toplama { // Dahili uye sinif
        public int toplamaYap(int a, int b) {
            return a + b;
        }
    } // class Toplama

    public static void main(String args[]) {
        Hesaplama.Toplama ht = new Hesaplama().new Toplama();
        int sonuc = ht.toplamaYap(3, 5);
        System.out.println("Sonuc = " + sonuc);
    }
} // class Hesapla
```



DAHİLİ ÜYE SINIFLAR VE ERİŞİM BELİRLEYİCİLERİ

- Dâhili üye sınıflara, public, friendly, protected veya private erişim belirleyicileri atanabilir, böylece dâhili üye sınıflarımıza olan erişimi kısıtlamış/açmış oluruz.
- Dikkat edilmesi gereken diğer bir husus ise bir dâhili üye sınıf **private** erişim belirleyicisine sahip olsa dahi, çevreleyici sınıf içerisindeki tüm metotlar tarafından erişilebilir olmasıdır.
- Eğer dahili üye sınıfımızı static olarak tanımlarsak, bu dahili sınıfın elamanlarında erişmek için *anasınıf.dahilisınıf* ile değil de doğrudan sınıf adı ile nesne oluştururuz.

Örnek(*Hesaplama1.java*)

```java
package Dahili_Siniflar;

public class Hesaplama1 {
    public class Toplama { // Dahili uye sinif - public
        public int toplamaYap(int a, int b) {
            return a + b;
        }
    } // class Toplama

    protected class Cikartma { // Dahili uye sinif - protected
        public int cikartmaYap(int a, int b) {
            return a - b;
        }
    } // class Cikartma

    class Carpma { // Dahili uye sinif - friendly
        public int carpmaYap(int a, int b) {
            return a * b;
        }
    } // class Carpma

    private class Bolme { // Dahili uye sinif - private
        public int bolmeYap(int a, int b) {
            return a / b;
        }
    } // class Bolme
```

```java

    public static void main(String args[]) {
        Hesaplama1.Toplama ht = new Hesaplama1().new Toplama();
        Hesaplama1.Cikartma hck = new Hesaplama1().new Cikartma();
        Hesaplama1.Carpma hcp = new Hesaplama1().new Carpma();
        Hesaplama1.Bolme hb = new Hesaplama1().new Bolme();
        int sonuc1 = ht.toplamaYap(10, 5);
        int sonuc2 = hck.cikartmaYap(10, 5);
        int sonuc3 = hcp.carpmaYap(10, 5);
        int sonuc4 = hb.bolmeYap(10, 5);
        System.out.println("Toplama Sonuc = " + sonuc1);
        System.out.println("Cikartma Sonuc = " + sonuc2);
        System.out.println("Carpma Sonuc = " + sonuc3);
        System.out.println("Bolme Sonuc = " + sonuc4);
    }
} // class Hesaplama
```

Console | Problems | Debug Shell

```
<terminated> Hesaplama1 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (28 Mar 202
Toplama Sonuc = 15
Cikartma Sonuc = 5
Carpma Sonuc = 50
Bolme Sonuc = 2
```

Örnek(*Hesaplama1.java*)

- Hesaplama1 sınıfımızın içerisinde toplam 4 adet dâhili üye sınıf mevcuttur. Public erişim belirleyicisine sahip Toplama dâhili üye sınıfı, protected erişim belirleyicisene sahip Cikartma dâhili üye sınıfı, friendly erişim belirleyicisine sahip Carpma dahili üye sınıfı ve private erişim belirleyicisine sahip Bolme dahili üye sınıfı. Hesaplama1 sınıfı, bu 4 adet dahili üye sınıfın çevreliyici sınıfıdır. Çevreleyici olan Hesaplama1 sınıfının statik olan main() yordamına dikkat edilirse, bu yordamın içerisinde tüm (private dâhil) dâhili üye sınıflara erişilebildiğini görülür. Bunun sebebi, main() yordamı ile tüm dâhili üye sınıfların aynı çevreleyici sınıfın içerisinde olmalarıdır.

Örnek(*Hesaplama2Kullan.java*)

```java
package Dahili_Siniflar;

class Hesaplama2 {
      public class Toplama2 { // Dahili uye sinif - public
         public int toplamaYap(int a, int b) {
            return a + b ;
         }
      } // class Toplama2

      protected class Cikartma2 { // Dahili uye sinif - protected
         public int cikartmaYap(int a, int b) {
            return a - b ;
         }
      } // class Cikartma2

      class Carpma2 { // Dahili uye sinif - friendly
         public int carpmaYap(int a, int b) {
            return a * b ;
         }
      } // class Carpma2

      private class Bolme2 { // Dahili uye sinif - private
         public int bolmeYap(int a, int b) {
            return a / b ;
         }
      } // class Bolme2

   } // class Hesaplama2
```

```java
public class Hesaplama2Kullan {
   public static void main(String args[]) {

        Hesaplama2.Toplama2 ht=new Hesaplama2().new Toplama2() ;
           Hesaplama2.Cikartma2 hck=new Hesaplama2().new Cikartma2() ;
           Hesaplama2.Carpma2 hcp = new Hesaplama2().new Carpma2() ;
           // Hesaplama2.Bolme3 hb = new Hesaplama2().new Bolme2() ;
           // ! Hata !

           int sonuc1 = ht.toplamaYap(10,5);
           int sonuc2 = hck.cikartmaYap(10,5);
           int sonuc3 = hcp.carpmaYap(10,5);
           // int sonuc4 = hb.bolmeYap(10,5); // ! Hata !

           System.out.println("Toplama Sonuc = " + sonuc1 );
           System.out.println("Cikartma Sonuc = " + sonuc2 );
           System.out.println("Carpma Sonuc = " + sonuc3 );
   }
}
```

Console | Problems | Debug Shell

```
<terminated> Hesaplama2Kullan [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (28 Mar 2020 22:36
Toplama Sonuc = 15
Cikartma Sonuc = 5
Carpma Sonuc = 50
```

Örnek(*Hesaplama2Kullan.java*)

- Normalde bir sınıf private veya protected erişim belirleyicisine sahip olamaz ancak dahili sınıflar private veya protected erişim belirleyicisine sahip olabilir. Hesaplama2Kullan sınıfı, Hesaplama2 sınıfı ile aynı paket içerisinde olduğu için, Hesaplama2Kullan sınıfı, Hesapla2 sınıfının içerisinde tanımlanmış olan public, protected ve friendly erişim belirleyicilerine sahip olan dahili üye sınıflara erişebilir ama private erişim belirleyicisine sahip olan Bolme dahili üye sınıfına erişemez.

Dâhili Üye Sınıflar ve Bunları Çevreleyen Sınıflar Arasındaki İlişki

- Dâhili üye sınıflar, içerisinde bulundukları çevreleyici sınıfların tüm alanlarına (statik veya değil private dâhil) ve metotlarına (statik veya değil-private dâhil) erişebilirler.

```java
package Dahili_Siniflar;

public class Hesaplama3 {
    private int sabit1 = 2;
    private static int sabit2 = 1;

    public class Toplama3 { // Uye dahili sinif
        public int toplamaYap(int a, int b) {
            return (a + b) + sabit1; // dikkat
        }
    } // class Toplama3

    public class Cikartma3 { // Uye dahili sinif
        public int cikartmaYap(int a, int b) {
            dekontBilgileriGoster(); // dikkat
            return (a - b) - sabit2; // dikkat
        }
    } // class Cikartma3

    private void dekontBilgileriGoster() {
        System.out.println("Dekont Bilgileri Gosteriliyor");
    }

    public void ekranaBas(int a, int b) {
        int sonuc = new Toplama3().toplamaYap(a, b);
        System.out.println("Sonuc = " + a + " + " + b + " + sabit1 = " + sonuc);
    }

    public static void main(String args[]) {

        Hesaplama3 h3 = new Hesaplama3();
        h3.ekranaBas(10, 5);

        // Toplama islemi
        Hesaplama3.Toplama3 ht3 = h3.new Toplama3();
        int sonuc = ht3.toplamaYap(11, 6);
        System.out.println("Sonuc = 11 + 6 + sabit1 = " + sonuc);

        // Cikartma islemi
        Hesaplama3.Cikartma3 hc3 = h3.new Cikartma3();
        int sonuc1 = hc3.cikartmaYap(10, 5);
        System.out.println("Sonuc = 10 - 5 - sabit2 = " + sonuc1);
    }
} // class Hesaplama3
```

```
Console  Problems  Debug Shell
<terminated> Hesaplama3 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (28 Mar 202
Sonuc = 10 + 5 + sabit1 = 17
Sonuc = 11 + 6 + sabit1 = 19
Dekont Bilgileri Gosteriliyor
Sonuc = 10 - 5 - sabit2 = 4
```

Örnek(*Hesaplama3.java*)

- Hesaplama3 sınıfının içerisinde iki adet dâhili üye sınıf bulunmaktadır. Bunlar Toplama3 ve Cikartma3 sınıflarıdır. Toplama3 dahili üye sınıfı, Hesaplama3 sınıfı içerisinde global olarak tanımlanmış ilkel (primitive) int tipindeki ve private erişim belirleyicisine sahip olan sabit1 alanına erişebilmektedir. Toplama3 dahili üye sınıfı, Hesaplama3 sınıfı içerisinde tanımlanmış olan sabit1 alanını kullanırken sanki kendi içerisinde tanımlanmış bir alanmış gibi, hiç bir belirteç kullanmamaktadır.
- Aynı şekilde Cikartma3 dâhili üye sınıfı, Hesaplama3 sınıfının içerisinde statik olarak tanımlanmış, private erişim belirleyicisine sahip ilkel int tipindeki sabit2 alanını ve private erişim belirleyicisine sahip dekontBilgileriGoster() yordamına direk olarak erişebilmektedir.

Örnek(*Hesaplama3.java*)

- Hesaplama3 sınıfının, nesne yordamı olan (-bu yordamın kullanılabilmesi için Hesaplama3 sınıfına ait bir nesne oluşturmak gerekir) ekranaBas(), iki adet parametre alıp, geriye hiçbirşey döndürmez (void). Bu yordamın içerinde Toplama3 dahili üye sınıfına ait nesne oluşturularak, bu dahili üye sınıfın toplamaYap() yordamı çağrılmaktadır. Toplama3 dâhili üye sınıfının toplamaYap() yordamından dönen cevap, ekranaBas() yordamının içerisinde ekrana bastırılır.
- Sadece bir adet Hesaplama3 sınıfına ait nesne oluşturuldu. Bu nesneye bağlı referansı kullanarak (h3), diğer dâhili üye sınıflara ait nesneler oluşturulabilir. Buradaki ana fikir, çevreleyici sınıfların içerisinde bulunan her dâhili üye sınıfa ait bir nesne oluşturmak için, her seferinde yeni bir çevreleyici sınıfa ait nesne oluşturma zorunluluğu olmadığıdır. Yani çevreleyici sınıfa ait bir nesne, yine çevreleyici sınıf tipindeki bir referansa bağlanırsa, işler daha kestirmeden çözülebilir.

Statik Dâhili Üye Sınıflar

- Statik (static) olarak tanımlanmış dâhili üye sınıflar, normal dahili üye sınıflardan farklıdırlar. Bu farklılıklar şöyledir:
  - Statik dâhili üye sınıfına ait nesne oluşturmak için, onu çevreleyen sınıfa ait bir nesne oluşmak zorunda değiliz.
  - Statik dâhili üye sınıflar, kendilerini çevreleyen sınıfa ait bağlantıyı (-this-) kaybederler.
    - Statik dahili üye sınıflar, onları çevreleyen üst sınıfa ait global alanlara (statik veya değil) ve yordamlara/metotlara (statik veya değil) direk ulaşım şansını kaybeder. Bunun sebebi, kendisini çevreleyen sınıf ile arasındaki bağı kopartmış olmasıdır.

Örnek(*Hesaplama4.java, Hesaplama4Kullan.java*)

```java
package Dahili_Siniflar;
public class Hesaplama4 {
   int sabit = 2;
   private int ozelsabit = 1 ;
   public static class Toplama4 { // Statik uye dahili sinif
      static int toplam ; // dogru
      int sonuc ; // dogru
      public int toplamaYap(int a, int b) {
         // return (a+b) + sabit ; ! Hata !
         sonuc = toplam = a+b;
         return sonuc;
      }
      public void dekontOlustur() {
         /* -sabit- alanina ve
         -ekranaBas() yordamına ulasabilmek icin
         Hesaplama4 sinifina ait nesne olusturmamiz gerekir.
         */
         Hesaplama4 hs4 = new Hesaplama4(); //dikkat
         int a = hs4.ozelsabit ; // dogru
         hs4.ekranaBas() ; //dogru
         System.out.println("Dekont olusturuyor = " +
         hs4.sabit + " - " +a );
      }
   } // class Toplama4
   public class Cikartma4 { //Uye dahili sinif
      int sonuc ;
      // static int sonuc1 ; // ! hata!
      public int cikartmaYap(int a, int b) {
         ekranaBas(); // dikkat
         sonuc = (a-b) - ozelsabit;
         return sonuc ; // dikkat
      }
   } // class Cikartma4
   private void ekranaBas() {
      System.out.println("Hesaplama4.ekranaBas()");
   }
```

```java
   public static void main(String args[]) {
      // ! Hata !
      // Hesaplama4.Toplama4 ht=new Hesaplama4().new Toplama4();
      Toplama4 tp4 = new Toplama4();
      tp4.dekontOlustur();
      int sonuc = tp4.toplamaYap(10,5);
      System.out.println("Sonuc = 10 + 5 = " + sonuc );
   }
} // class Hesaplama4
class Hesaplama4Kullan {
   public static void main(String args[]) {

      // ! Hata!
      // Hesaplama4.Toplama4 ht=new Hesaplama4().new Toplama4() ;
      Hesaplama4.Toplama4 tp4 = new Hesaplama4.Toplama4();
      int sonuc = tp4.toplamaYap(10,5);
      System.out.println("Sonuc = 10 + 5 = " + sonuc );
   }

} // class Hesaplama4Kullan
```

Hesaplama4.java

- **Toplama4** statik dahili sınıfının içerisinde statik global alan tanımlayabiliriz. Statik olmayan dahili üye sınıfların içerisinde statik global alan tanımlanamaz.
- Toplama4 statik dahili üye sınıfının, toplamaYap() yordamının içerisinde, Hesaplama4 sınıfına ait global olarak tanımlamış ilkel (primitive) int tipindeki sabit alanına direk erişilemez. Statik dâhili üye sınıflar ile bunları çevreleyen sınıflar arasında this bağlantısı yoktur. Eğer statik dâhili üye sınıfın içerisinden, onu çevreleyen sınıfa ait bir alan (statik olmayan) veya yordam (statik olmayan) çağrılmak isteniyorsa, bu bizzat ifade edilmelidir. Aynı Toplama4 statik dâhili üye sınıfına ait dekontOlustur() yordamının içerisinde yapıldığı gibidir.

- dekontOlustur() yordamının içerisinde, Hesaplama4 sınıfına ait nesne oluşturulmadan, sabit, ozelsabit alanlarına ve ekranaBas() yordamına ulaşamazdık. Buradaki önemli nokta, dâhili üye sınıf statik olsa bile, kendisine çevreleyen sınıfın private erişim belirleyicisi sahip olan alanlarına (statik veya değil) ve yordamlarına (statik veya değil) erişebilmesidir.

- Hesaplama4 sınıfının statik olan main() yordamının içerisinde, Toplama4 statik dâhili üye sınıfına ait nesnenin nasıl oluşturulduğuna dikkat edelim. Toplama4 statik dâhili üye sınıfına ait nesne oluştururken, onu çevreleyen sınıfa ait herhangi bir nesne oluşturmak zorunda kalmadık.

Hesaplama4Kullan.java

- Başka bir sınıfın içerisinde statik dâhili üye sınıfı ulaşmak için, sadece tanımlama açısından, dahili üye sınıfı çevreleyen sınıfın ismi kullanılmıştır. Mantıklı olanda budur, statik de olsa sonuçta ulaşılmak istenen dâhili üye bir sınıfıdır.

Statik Dâhili Üye Sınıflar ve Statik Metotlar

- Statik dâhili üye sınıfların içerisinde statik alanlar bulunduğu gibi, statik metotlarda bulunabilir. Eğer statik dâhili üye sınıfı içerisinde, statik bir metot oluşturulmuş ise, bu metodu çağırmak için ne statik dâhili üye sınıfına ne de onu çevreleyen sınıfa ait herhangi bir nesne oluşturmak gerekmez.

Örnek(*Hesaplama5.java*)

```java
package Dahili_Siniflar;

public class Hesaplama5 {
    private static int x = 3;

    public static class Toplama5 { // Statik uye dahili sinif
        static int toplam; // dogru
        int sonuc; // dogru

        public static int toplamaYap(int a, int b) {
            // sonuc = a+b + x ; // ! Hata !
            toplam = a + b + x;
            return toplam;
        }
    } // class Toplama5

    public static void main(String args[]) {
        int sonuc = Hesaplama5.Toplama5.toplamaYap(16, 8); // dikkat
        System.out.println("Sonuc = 16 + 8 = " + sonuc);
    }
} // class Hesaplama5
```

```
Console  Problems  Debug Shell
<terminated> Hesaplama5 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (28 Mar 2020 2
Sonuc = 16 + 8 = 27
```

- Toplama5 statik dâhili üye sınıfının, statik olan toplamaYap() yordamından, Hesaplama5 çevreleyici sınıfına ait ilkel (primitive) int tipinde tanımlanmış x alanına ulaşılabilir. Bunun sebebi x alanında statik olarak tanımlanmış olmasıdır. main() yordamının içerisinde, toplamaYap() yordamının çağrılışına dikkat edilirse, ne Hesaplama5 sınıfına ait nesne, ne de Toplama5 statik dâhili üye sınıfına ait bir nesnenin oluşturulmadığı görülür.

Statik ve Final Alanlar

- Statik olmayan dâhili üye sınıfların içerisinde, statik alanlar ve yordamlar/metotlar tanımlanamaz; ama "statik ve final" alanlar tanımlanabilir. Bir alanın hem statik hem de final olması demek, onun **sabit** olması anlamına geldiği için, Statik olmayan dâhili üye sınıfların içerisinde statik ve final alanlar kullanılabilir.

```
class CevreliyiciSinif1 {

  class DahiliSinif1 { // Dahili uye siniflar
    // static int x = 10 ; // ! Hata !
  }
}
// Dogru
class CevreliyiciSinif2 {

  class DahiliSinif2 {
    int x; // Dogru
  }
}
// Dogru
class CevreliyiciSinif3 {

  class DahiliSinif3 {
    static final int x = 0; // Dogru
  }
}
```

Dahili Üye Sınıflar ve Yapılandırıcılar (Constructors)

- Dâhili üye sınıfların yapılandırıcıları olabilir.

```
public class BuyukA {

  public class B {
    public B() { // yapilandirici
      System.out.println("Ben B sinifi ");
    }
  } // class B

  public BuyukA() {
    System.out.println("Ben BuyukA sinifi ");
  }

  public static void main(String args[]) {
    BuyukA ba = new BuyukA();
  }
}
```

İç içe Dahili Üye Sınıflar

- Bir sınıfın içerisinde dâhili üye sınıf tanımlayabilirsiniz. Tanımlanan bu dâhili üye sınıfın içerisinde, yine bir dâhili üye sınıf tanımlayabilirsiniz

```
public class Abc {

  public Abc() { // Yapilandirici
    System.out.println("Abc nesnesi olusturuluyor");
  }

  public class Def {
    public Def() { // Yapilandirici
      System.out.println("Def nesnesi olusturuluyor");
    }

    public class Ghi {
      public Ghi() { // Yapilandirici
        System.out.println("Ghi nesnesi olusturuluyor");
      }

    } // class Ghi

  } //class Def

  public static void main( String args[] ) {
    Abc.Def.Ghi ici_ice = new Abc().new Def().new Ghi();
  }

} // class Abc
```

Soyut (Abstract) Dâhili Üye Sınıflar

- Dâhili üye sınıflar, soyut (abstract) sınıf olarak tanımlanabilir. Bu soyut dâhili üye sınıflardan türeyen sınıflar, soyut dâhili üye sınıfların içerisindeki gövdesiz (soyut) yordamları/metotları iptal etmeleri gerekmektedir.
- Hayvan sınıfının içerisinde soyut (abstract) dâhili üye sınıf olarak tanımlanmış Kus sınıfı iki adet gövdesiz (soyut-abstract) yordamı olsun, uc() ve kon(). Kartal sınıfı, soyut dâhili üye sınıf olan Kus sınıfından türetilebilir.

Örnek(*Kartal.java*)

- Kartal sınıfının içerisinde, soyut dahili üye sınıf olan Kus sınıfının, gövdesiz olan iki yordamı iptal edilmiştir. Olayları sırası ile inceleyelim, Kartal sınıfına ait bir nesne oluşturulmak istense bunun öncesinde Kus sınıfına ait bir nesnenin oluşturulması gerekir çünkü Kartal sınıfı Kus sınıfından türetilmiştir. Buraya kadar sorun yok, fakat asıl kritik nokta Kus sınıfının dâhili üye sınıf olmasıdır. Daha açık bir ifade ile, eğer Kus sınıfına ait bir nesne oluşturulacaksa, bunun öncesinde elimizde Kus sınıfının çevreleyici sınıfı olan Hayvan sınıfına ait bir nesne bulunması zorunluluğudur. Kus sınıfı statik dahili üye sınıf olmadığından, Hayvan sınıfına bağımlıdır.

```
1  package Dahili_Siniflar;
2
3  class Hayvan {
4      abstract class Kus {
5          public abstract void uc();
6          public abstract void kon();
7      }
8      public void avlan() {
9          System.out.println("Hayvan avlaniyor...");
10     }
11 }
12 class Kartal extends Hayvan.Kus {
13     public void uc() {
14         System.out.println("Kartal Ucuyor...");
15     }
16     public void kon() {
17         System.out.println("Kartal Konuyor...");
18     }
19     // public Kartal() { } // ! Hata !
20
21     public Kartal(Hayvan hv) {
22         hv.super(); // Dikkat
23     }
24
25     public static void main(String args[]) {
26         Hayvan h = new Hayvan(); // Dikkat
27         Kartal k = new Kartal(h);
28         k.uc();
29         k.kon();
30     }
31 }
```

```
Console  Problems  Debug Shell
<terminated> Kartal [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (29 Mar 202
Kartal Ucuyor...
Kartal Konuyor...
```

2. YEREL SINIFLAR (LOCAL CLASSES)

- Yerel sınıflar, yapılandırıcıların (constructor), sınıf yordamlarının (statik yordam), nesne yordamların, statik alanlara toplu değer vermek için kullandığımız statik bloğun veya statik olmayan alanlara toplu değer vermek için kullandığımız bloğun içerisinde tanımlanabilir.

```
public class Sinif {
  public void yordam() {
    public class YerelSinif {
      //...
    }
  }
}
```

- Yerel sınıflar, yalnızca içinde tanımlandıkları, yordamın-metotun veya bloğun içerisinde geçerlidir. Nasıl ki dâhili üye sınıfların çevreleyici sınıfları vardı, yerel sınıfların ise çevreleyici yordamları veya blokları vardır. Yerel sınıflar tanımlandıkları bu yordamların veya blokların dışarısından erişilemezler.
  - Yerel sınıflar tanımlandıkları yordamın veya bloğun dışından erişilemezler.
  - Yerel sınıflar başka sınıflardan türetilebilir veya arayüzlere (interface) erişebilir
  - Yerel sınıfların yapılandırıcıları olabilir.
  - Yerel sınıflar, içinde bulundukları yordamın sadece final olan değişkenlerine ulaşabilirler.
  - Yerel sınıflar, statik veya statik olmayan yordamların içerisinde tanımlanabilirler.
  - Yerel sınıflar, private, protected ve public erişim belirleyicisine sahip olamazlar sadece friendly erişim belirleyicisine sahip olabilirler.
  - Yerel sınıflar, statik olarak tanımlanamaz.

Örnek(*Hesaplama6.java*)

```java
package Arayuzler;

interface Toplayici {
    public int hesaplamaYap();
}

public class Hesaplama6 {

    public int topla(int a, int b) {
        class Toplama6 implements Toplayici {
            private int deger1;
            private int deger2;

            public Toplama6(int deger1, int deger2) { // yapilandirici
                this.deger1 = deger1;
                this.deger2 = deger2;
            }

            public int hesaplamaYap() { // iptal etti (override)
                return deger1 + deger2;
            }

        } // class Toplama6

        Toplama6 t6 = new Toplama6(a, b);
        return t6.hesaplamaYap();
    }

    public void ekranaBas() {
        // Toplama6 t6 = new Toplama6(2,6,); // !Hata!-Kapsama alanının dısı
    }

```

```java

    public static void main(String args[]) {
        Hesaplama6 h6 = new Hesaplama6();
        int sonuc = h6.topla(5, 9);
        System.out.println("Sonuc = 5 + 9 = " + sonuc);
    }
} // class Hesaplama6
```

Console Problems Debug Shell

<terminated> Hesaplama6 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (29 Mar 2020 22

```
Sonuc = 5 + 9 = 14
```

Örnek(*Hesaplama6.java*)

- Bu örneğimizde Toplama6 yerel sınıftır. Yerel bir sınıf, başka bir sınıftan türetilebilir veya bir arayüze erişip, onun gövdesiz yordamlarını iptal edebilir, aynı normal sınıflar gibi. Toplama6 yerel sınıfı, Hesapliyici arayüzüne eriştiğinden, bu arayüzün gövdesiz yordamı olan hesaplamaYap() yordamını iptal etmek zorundadır. Toplama6 yerel sınıfı, Hesaplama6 sınıfının topla() yordamının içerisinde tanımlanmıştır. Bunun anlamı, Toplama6 yerel sınıfına yalnızca topla() yordamının içerisinde erişilebileceğidir. Hesaplama6 sınıfının nesne yordamı olan (bu yordama ulaşmak için Hesaplama6 sınıfına ait nesne oluşturmamız gerektiği anlamında...) ekranaBas() yordamının içerisinden, Toplama6 yerel sınıfına ulaşılamaz çünkü Toplama6 yerel sınıfı, ekranaBas() yordamının kapsama alanının dışında kalmaktadır.

3. İSİMSİZ SINIFLAR (ANONYMOUS CLASSES)

- İsimsiz sınıflar, isimsiz ifade edilebilen sınıflardır. İsimsiz sınıflar havada oluşturulabildiklerinden dolayı birçok işlem için çok avantajlıdır, özellikle olay dinleyicilerin (event listeners) devreye sokulduğu uygulamalarda sıkça kullanılırlar. İsimsiz sınıfların özellikleri aşağıdaki gibidir;
  - Diğer dâhili sınıf çeşitlerinde olduğu gibi, isimsiz sınıflar direk extends ve implements anahtar kelimelerini kullanarak, diğer sınıflardan türetilemez ve arayüzlere erişemez.
  - İsimsiz sınıfların herhangi bir ismi olmadığı için, yapılandırıcısında (constructor) olamaz.
- Not: *Program çalışırken kullanıcı tarafından programa dair gerçekleştirilen tüm hareketlere **olay(event)** denir. Örneğin kullanıcının programda bir buttona tıklaması, fareyi hareket ettirmesi, programı kapatması vb. şekilde kullanıcının programla etkileşimi sonucunda olay(event) meydana gelir.*

Örnek(Hesaplama7.java)

- Hesaplama7 sınıfının, topla() yordamı Toplayici arayüzü tipindeki nesneye bağlı bir referans geri döndürmektedir. Toplayici arayüzü tipindeki nesneye bağlı bir referans geri döndürmek demek, Toplayici arayüzüne erişip onun gövdesiz olan yordamlarını iptal eden bir sınıf tipinde nesne oluşturmak demektir. Sonuçta bir arayüze ulaşan sınıf, ulaştığı arayüz tipinde olan bir referansa bağlanabilirdi.

- **Not:** *İsimsiz metottan sonra noktalı virgül kullanılmasının sebebi,* ***return*** *kelimesinden sonra tanımladığımız isimsiz metodumuzun bir değer olarak algılanmasını sağlamaktır.*

```
1  package Dahili_Siniflar;
2
3  interface Toplayiciii {
4          public int hesaplamaYap() ;
5      }
6
7  public class Hesaplama7 {
8
9      public Toplayiciii topla(final int a, final int b) {
10         return new Toplayiciii() {
11             public int hesaplamaYap() {
12
13                 // final olan yerel degiskenlere ulasabilir.
14                 return a + b;
15             }
16         }; // noktali virgul sart
17
18     } // topla, yordam sonu
19
20     public static void main(String args[]) {
21
22         Hesaplama7 h7 = new Hesaplama7();
23         Toplayiciii t = h7.topla(5, 9);
24         int sonuc = t.hesaplamaYap();
25         System.out.println("Sonuc = 5 + 9 = " + sonuc);
26     }
27 } // class Hesaplama7
```

```
Console  Problems  Debug Shell
<terminated> Hesaplama7 [Java Application] C:\Program Files\Java\jdk-13.0.1\bin\javaw.exe (29 Mar 2020 22:24:4
Sonuc = 5 + 9 = 14
```

- **Not:** *İsimsiz metottan sonra noktalı virgül kullanılmasının sebebi,* ***return*** *kelimesinden sonra tanımladığımız isimsiz metodumuzun bir değer olarak algılanmasını sağlamaktır.*

Ödev

- Klavyeden girilen Stringin kaç adet karakter içerdiğini, büyük halini ve küçük halini ekrana yazdıran sınıfı yazın. Bunları ekrana yazdıracak metotlardan biri dahili üye sınıfın metodu, diğeri yerel sınıfın metodu, bir sonraki ise isimsiz sınıfın metodu olsun.

Kaynaklar

- [1]. http://mail.baskent.edu.tr/~tkaracay/etudio/ders/prg/java/ch17/interface.html
- [2]. http://javayaz.com/?page_id=2048
- [3]. Kirazlı, M., Tanrıverdi, S. JAVA Yeni Başlayanlar İçin. Kodlab. Baskı 8.
- [4]. http://www.csharpnedir.com/articles/read/?id=846